

**Sample Test SY-SOC S14-ST**


Time 120 minutes  
– use of class documents allowed –

Name: \_\_\_\_\_

Percent: \_\_\_\_\_

Matr. No.: \_\_\_\_\_

Grade: \_\_\_\_\_

|   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |  | Σ |
|---|---|---|---|---|---|---|

**(1) General SoC Questions**

- (1.1) – RISC / CISC Architecture
- Memory types
  - Floating and fixed-point arithmetic
  - I-Cache / D-Cache (reasons, benefits)
  - Pipelining architecture (pros / cons; branch prediction)
  - Interrupt mechanism (hardware, software)
  - FPGA integrated CPUs, soft cores (i.e. PicoBlaze, MicroBlaze)

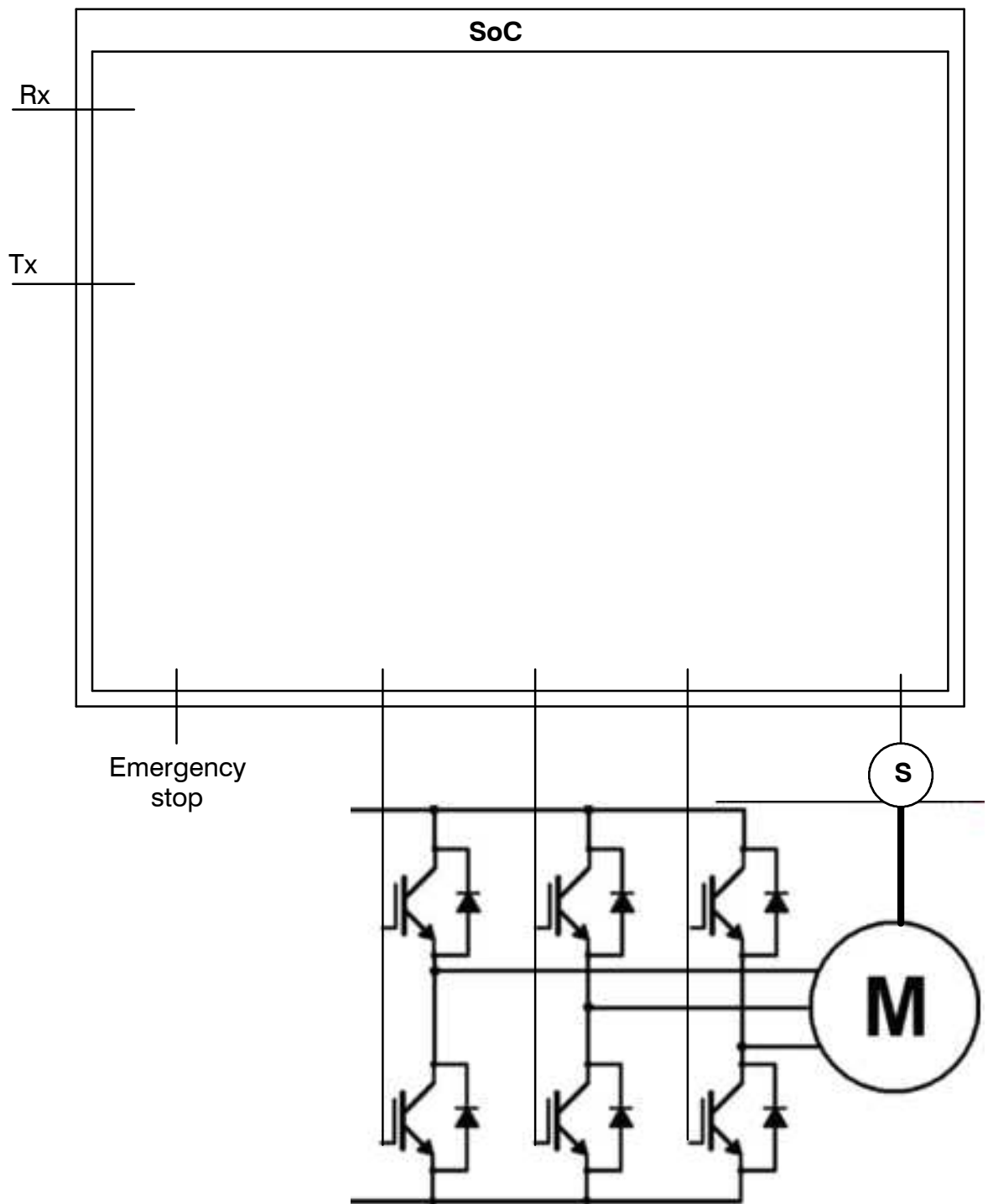
**(2) SoC Block Diagram for an Industrial Control System**

(2.1) Complete the following block diagram for an industrial AC drive control system.  
The diagram should contain:

- Required user logic blocks
- IP block (available logic)
- Processor
- bus systems
- signal connections

|          |          |          |          |          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| (1) = 5  | (2) = 6  | (3) = 6  | (4) = 7  | (5) = 7  | Σ = 31   |          |          |          |          |
| 1.0 ≥ 29 | 1.3 ≥ 27 | 1.7 ≥ 26 | 2.0 ≥ 24 | 2.3 ≥ 23 | 2.7 ≥ 21 | 3.0 ≥ 20 | 3.3 ≥ 18 | 3.7 ≥ 17 | 4.0 ≥ 15 |

**SY-SOC / ESD 2014 sample test**



**(3) DSP**

---

State feedback control has to be designed with fixed point arithmetic. The state feedback coefficients are given by

$$F = [ 4.231 \quad -7.323 \quad 14.2 \quad 9.43 ] .$$

For the coefficients 14 bits are available; data is stored in 10 bit words.

- (3.1) Select optimal fixed point format for  $F$  (maximum precision, maximum possible number of fractional bits). What are the possible minimum and maximum values for this format?
- (3.2) What is the decimal weight for each bit position?
- (3.3) Draw a *detailed* block diagram for the product  $u = Fx$  and write down the fixed point formats after each sum and each product (enough digits to prevent overflows).

**(4) SoC Hardware**

---

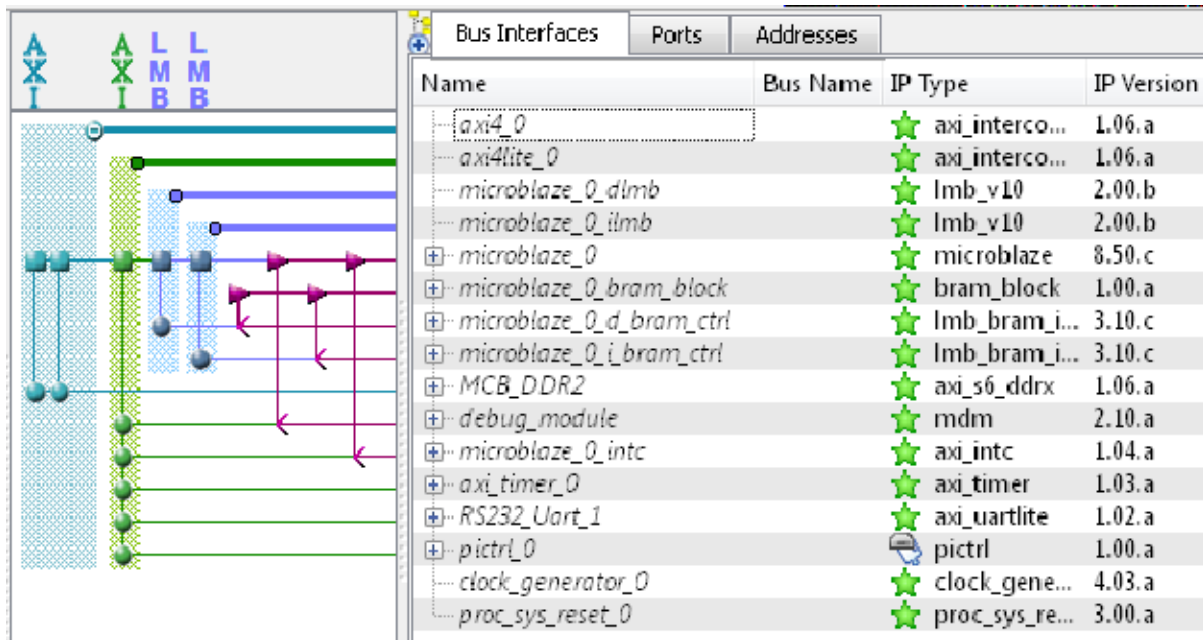
PI Controller in hardware is required for performance reasons. A user logic AXI block for hardware solution of the PI controller should be designed. The PI controller parameters (proportional and integral gain) are:

$$p\_gain = 0.8 \text{ and } i\_gain = 0.15 .$$

Word size of 16 bits should be used for coefficients and for data (signed fixed point integer).

---

- (4.1) Calculate the fixed point values for  $p\_gain$  and  $i\_gain$  in int16.14 format.
- (4.2) Analyze the hardware system according to the following figure:



- (4.3) Assume that you created a `userlogic.vhd` file with register support for hardware PI controller block. Moreover the library `ieee.std_logic_arith.all` has been included to allow for conversion between `std_logic_vector` and `integer` (and subtypes of `integer`).

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;
```

Hints:

- a) conversion from `std_logic_vector` to `integer`:

```
conv_integer(signed(<std_logic_vector_signal>))
```

- b) conversion from `integer` to `std_logic_vector`:

```
conv_std_logic_vector(<integer_var>, <n_bits>)
```

Write the `userlogic.vhd` file for PI controller hardware calculation. Take care that a start condition exists to calculate one discrete step. Input and output data is transferred by registers.

- (4.4) Explain the major elements which carry out the algorithm. Use the supplied synthesis report below:

```

=====
HDL Synthesis Report

Macro Statistics
# Multipliers                               : 2
  16x13-bit multiplier                       : 1
  16x15-bit multiplier                       : 1
# Adders/Subtractors                         : 2
  32-bit adder                               : 2
# Registers                                  : 2
  32-bit register                            : 2
# FSMs                                        : 1
=====

Device utilization summary:
-----

Selected Device : 6slx45csg324-3

Slice Logic Utilization:
Number of Slice Registers:      2 out of 54576    0%
Number of Slice LUTs:          3 out of 27288    0%
  Number used as Logic:        3 out of 27288    0%

Slice Logic Distribution:
Number of LUT Flip Flop pairs used: 5
  Number with an unused Flip Flop: 3 out of 5    60%
  Number with an unused LUT:       2 out of 5    40%
  Number of fully used LUT-FF pairs: 0 out of 5    0%
  Number of unique control sets:   1

IO Utilization:
Number of IOs:                  99
Number of bonded IOBs:         83 out of 218    38%

Specific Feature Utilization:
Number of BUFG/BUFGCTRLs:      1 out of 16     6%
Number of DSP48A1s:            2 out of 58     3%

```

## (5) SoC Software

The software side for the PI controller hardware should be written.

- (5.1) Assume that `xparameters.h` contains the base address for the PI controller hardware:

```
/* Definitions for peripheral PICTRL_0 */  
#define XPAR_PICTRL_0_BASEADDR 0x76400000
```

Write the driver for your PI controller hardware from (4) producing a similar output as below:

```
-- PI (Hardware) Test V0.0a ---
```

```
=> s
```

```
[integr] = 2458
```

```
[pi_out] = 13107
```

```
=> s
```

```
[integr] = 4916
```

```
[pi_out] = 15565
```

```
=> s
```

```
[integr] = 7374
```

```
[pi_out] = 18023
```

```
=> s
```

```
[integr] = 9832
```

```
[pi_out] = 20481
```

```
=> s
```

```
[integr] = 12290
```

```
[pi_out] = 22939
```

```
=> s
```

```
[integr] = 14748
```

```
[pi_out] = 25397
```

```
=> s
```

```
[integr] = 17206
```

```
[pi_out] = 27855
```

```
=> s
```

```
[integr] = 19664
```

```
[pi_out] = 30313
```

```
=> x
```

```
[integr] = 22122
```

```
[pi_out] = 32771
```

```
Thank you for using PI.
```

Is the output reasonable?

---

(5.2) Analyze the program code and data sections according to address map and the elf header listing.

| Instance                   | Base Name        | Base Address | High Address | Size | Bus Interface(s) |
|----------------------------|------------------|--------------|--------------|------|------------------|
| microblaze_0's Address Map |                  |              |              |      |                  |
| microblaze_0_d_bram_ctrl   | C_BASEADDR       | 0x00000000   | 0x00001FFF   | 8K   | SLMB             |
| microblaze_0_i_bram_ctrl   | C_BASEADDR       | 0x00000000   | 0x00001FFF   | 8K   | SLMB             |
| RS232_Uart_1               | C_BASEADDR       | 0x40600000   | 0x4060FFFF   | 64K  | S_AXI            |
| microblaze_0_intc          | C_BASEADDR       | 0x41200000   | 0x4120FFFF   | 64K  | S_AXI            |
| debug_module               | C_BASEADDR       | 0x41400000   | 0x4140FFFF   | 64K  | S_AXI            |
| axi_timer_0                | C_BASEADDR       | 0x41C00000   | 0x41C0FFFF   | 64K  | S_AXI            |
| pictrl_0                   | C_BASEADDR       | 0x76400000   | 0x7640FFFF   | 64K  | S_AXI            |
| MCB_DDR2                   | C_S0_AXI_BASE... | 0xA8000000   | 0xAFFFFFFF   | 128M | S0_AXI           |

```
pitest.elf:      file format elf32-microblazeel
```

#### Sections:

| Idx | Name                  | Size     | VMA      | LMA      | File off | Algn |
|-----|-----------------------|----------|----------|----------|----------|------|
| 0   | .vectors.reset        | 00000008 | 00000000 | 00000000 | 00000094 | 2**2 |
| 1   | .vectors.sw_exception | 00000008 | 00000008 | 00000008 | 00000008 |      |
|     |                       | 0000009c |          |          |          | 2**2 |
| 2   | .vectors.interrupt    | 00000008 | 00000010 | 00000010 | 000000a4 |      |
| 2   |                       |          |          |          |          | **2  |
| 3   | .vectors.hw_exception | 00000008 | 00000020 | 00000020 | 00000020 |      |
|     |                       | 000000b4 |          |          |          | 2**2 |
| 4   | .text                 | 000013b8 | a8000000 | a8000000 | 000000bc | 2**2 |
| 5   | .init                 | 0000003c | a80013b8 | a80013b8 | 00001474 | 2**2 |
| 6   | .fini                 | 00000020 | a80013f4 | a80013f4 | 000014b0 | 2**2 |
| 7   | .ctors                | 00000008 | a8001414 | a8001414 | 000014d0 | 2**2 |
| 8   | .dtors                | 00000008 | a800141c | a800141c | 000014d8 | 2**2 |
| 9   | .rodata               | 00000456 | a8001424 | a8001424 | 000014e0 | 2**2 |
| 10  | .sdata2               | 00000006 | a800187a | a800187a | 00001936 | 2**0 |
| 11  | .data                 | 00000128 | a8001880 | a8001880 | 00001938 | 2**2 |
| 12  | .bss                  | 00000020 | a80019a8 | a80019a8 | 00001a60 | 2**2 |
| 13  | .heap                 | 00000400 | a80019c8 | a80019c8 | 00001a60 | 2**0 |
| 14  | .stack                | 00000400 | a8001dc8 | a8001dc8 | 00001a60 | 2**0 |

What could be done to speed up program execution (Hint: linking progress)?

